Lab 4: LCD Display

Introduction

In lab 4, you'll use everything you have learned so far to interface the TM4C LaunchPad with a liquid crystal display (LCD). With the LCD, you'll display Analog to Digital Converter (ADC) readings, create graphics, and use it to create a simple graphical user interface for your traffic light controller.

By the end of this lab, you will:

- Be even more familiar with the use of timers and ADCs with the TM4C123.
- Know how to interface with the LCD to display information and graphics.
- Become even better at extracting information from a datasheet

Liquid Crystal Display

LCD Hardware

For this lab, we're using the Kentec Display EB-LM4F120-L35 touchscreen LCD. It's a 3.5″ LCD "boosterpack" designed to be used with Stellaris LaunchPads, with a resolution of 320×240 pixels.

To begin, plug the LCD female headers into the male headers of your TIVA board, as shown in Figure 1.

Figure 1: LCD connected to the TIVA LaunchPad

Note: To use the LCD with the LaunchPad, the resistors R9 and R10 on the TIVA board need to be removed. This should already have been done for you but be sure to check before plugging it in – they're located near the center chip. If the resistors have not been removed, do not try to remove them yourself. Instead, bring your board to your TA.

The LCD interface uses pins 2-7 of port A, 0-7 of port B, and 4-5 of port E.

LCD Software

On the course's canvas webpage, you will find files to drive the LCD display. There's a lot of code there – some of it you'll need to fill in, some of it you'll use, some of it you won't use, and some of it doesn't work. How exciting!

For the first part of the lab, you'll be working with and adding to the device driver code for the LCD. A device driver contains function definitions and appropriate I/O port mappings to ensure proper functionality of the device, with the goal to create an easy-to-use interface between the device and other systems.

Task 1 – LCD Driver

Your first task is to get the LCD driver and display working. To begin, download SSD2119.c and SSD2119.h from the course Canvas webpage.

In the .c file, fill in the section marked "TODO" based on the comments found there. This will be just like previous GPIO port initializations.

Note 1: Make sure to include tm4c123gh6pm.h in the path indicated at the top of SSD2119.c

Note 2: The "TODO" code is called by the `LCD_Init()` function. This function is very long, and is sourced from Kentec Display. You don't need to know any of the details of what happens here.

Note 3:

The LCD_GPIOInit (initialization) will be similar to the previous port initializations: activate the clock, configure the direction register, turn off alternate functions. If you follow the comments we provided to you in the driver file, you'll see a couple wait for port activation. This is because the GPIO modules use clock gating to save power (we discussed this in the lecture). When they are enabled, the gated clock needs a few system clock cycles to be synchronized and fully stable.

According to Page 659 of the Datasheet, after the GPIO module clock is enabled, there must be a delay of 3 system clocks before GPIO registers can be accessed. In the LCD_GPIOInit function, this is done by spending one cycle to initialize the port, plus 2 wait++ statements, which takes 1 system clock cycle each.

Once you have completed the `LCD_GPIOInit()` function, you should test to see if everything is working. In `main()`, call `LCD_Init()`, and then call `LCD_ColorFill()` with a color of your choice. Running the code, you should see the screen filled with color. If it doesn't, try power cycling the board.

Task 2 – Temperature Readings

Previously, you displayed the temperature of your TM4C over UART to a terminal in PuTTy. Now, display the temperature on the LCD display.

Print the temperature readings as a floating point number, displaying at least two decimal places. You don't need to do any of the other requirements from lab 3, such as changing the clock speed.

Note: `LCD_PrintFloat()` *doesn't do what you want it to.*

Task 3 – Traffic Light

Recall the timed traffic light from lab 2. Repeat this task, but instead of using external LEDs, create virtual traffic lights on the LCD display. The buttons and the rest of the traffic light system should have the same functionality as in task 1 of lab 2.

For a badge, instead of using the physical buttons, create virtual buttons using the touch-screen functionality of the LCD. Label the buttons appropriately, and make sure that they work the same way as the physical buttons.

Lab Demonstration

Demonstrate your design for the assigned tasks to the TA.

Submission

Submit a short lab report that includes the following sections:

1. Procedure
   - Describe how you approached the assigned tasks and include any relevant diagrams, schematics, etc.
2. Results
   - Describe the results obtained for the assigned tasks. Give a brief overview of the finished product, as it relates to what was asked of you.
   - Include any relevant images, screenshots, etc.
3. Problems Encountered and Feedback
   - Describe any issues you had while completing the lab, and how/if you were able to overcome them.
   - Include any feedback you have about the lab, or any tips and tricks that you learned that may be useful to you in future labs.
   - Include how many hours you spent in completing this lab.
4. Appendix
   - In your Appendix, include any code you wrote in the lab. This should be **formatted** and **commented.**

   A good resource for formatting code for Word documents is found here: http://www.planetb.ca/syntax-highlight-word

Additionally, submit your commented .c and .h files.

Grading

Your grade for this assignment will depend on the following:

- Lab demonstration: **50%**

    - Your lab demonstration will be graded based on your completion of the assigned tasks, your ability to answer questions, and your understanding of the course material.
- Lab report: **20%**
    - Your lab report should be short and concise.
- Code comments: **30%**
    - Comment your code! Your code should have function comments and in-line comments where appropriate.